An Evolutionary Random Search Algorithm for Solving Markov Decision Processes

*by Jiaqiao Hu, Michael C. Fu, Vahid Ramezani,
and Steven I. Marcus*

TR 2005-3

An Evolutionary Random Policy Search Algorithm for Solving Markov Decision Processes

Jiaqiao Hu

Department of Electrical and Computer Engineering & Institute for Systems Research,
University of Maryland, College Park, MD 20742, jqhu@glue.umd.edu

Michael C. Fu

Robert H. Smith School of Business & Institute for Systems Research,
University of Maryland, College Park, MD 20742, mfu@rhsmith.umd.edu

Vahid R. Ramezani

Institute for Systems Research,
University of Maryland, College Park, MD 20742, rvahid@isr.umd.edu

Steven I. Marcus

Department of Electrical and Computer Engineering & Institute for Systems Research,
University of Maryland, College Park, MD 20742, marcus@eng.umd.edu

March 2004; revised September 2004

This paper presents a new randomized search method called Evolutionary Random Policy Search (ERPS) for solving infinite horizon discounted cost Markov Decision Process (MDP) problems. The algorithm is particularly targeted at problems with large or uncountable action spaces. ERPS approaches a given MDP by iteratively dividing it into a sequence of smaller, random, sub-MDP problems based on information obtained from random sampling of the entire action space and local search. Each sub-MDP is then solved approximately by using a variant of the standard policy improvement technique, where an elite policy is obtained. We show that the sequence of elite policies converges to an optimal policy with probability one. An adaptive version of the algorithm that improves the efficiency of the search process while maintaining the convergence properties of ERPS is also proposed. Some numerical studies are carried out to illustrate the algorithm and compare it with existing procedures.

Keywords: Markov decision process (MDP), policy iteration (PI), evolutionary policy iteration (EPI), genetic algorithms (GAs), global optimization.

1 Introduction

From operations research to artificial intelligence, a broad range of problems can be formulated and described by Markov decision process (MDP) models. Up to now, the majority of the solution methods have concentrated on reducing the size of the state space in order to address the well-known "curse of dimensionality" (i.e., the typical exponential growth in the state space size with the parameters of the problem). Some well-established approaches are state aggregation (Bertsekas and Castañon 1989), feature extraction (Tsitsiklis and Van Roy 1996), random successive approximations and random multigrid algorithms (Rust 1997), and various value function approximation schemes via the use of basis functions (de Farias and Van Roy 2003, Trick and Zin 1997), just to name a few. The key idea throughout is to avoid enumerating the entire state space. However, most of the above approaches generally require the ability to search the entire action space in order to choose the best action at each step of the iteration procedure; thus problems with very large action spaces may still pose a computational challenge.

The approach proposed in this paper is meant to complement these highly successful techniques. In particular, we will focus on MDPs where the state space is relatively small but the action space is very large, so that enumerating the entire action space becomes practically inefficient. These types of problems often arise in control of queues. For example, consider the problem of controlling the service rate of a single-server queue with a finite buffer size, say M, in order to minimize the average number of jobs in queue and the service cost. The state space of this problem is the possible number of jobs in the queue $\{0, 1, \ldots, M\}$, so the size of the state space is $M+1$, whereas the possible actions might be all values on an interval (e.g., representing a service rate), in which case the action space is uncountable. From a more general point of view, if one of the aforementioned state space reduction techniques is considered, for instance, say state aggregation, then MDPs with small state spaces and large action spaces can also be regarded as the outcomes resulting from the aggregation of MDPs with large state and action spaces.

The issue of large action spaces was addressed in early work by MacQueen (1966), who used some inequality forms of Bellman's equation together with bounds on the optimal value function to identify and eliminate non-optimal actions in order to reduce the size of the action sets to be searched at each iteration of the algorithm. Since then, the procedure has been applied to several standard methods like policy iteration (PI), value iteration (VI) and modified policy iteration (cf. e.g., Puterman 1994 for a review). All of these algorithms generally require that the admissible set of actions at each state is finite. In this paper, we approach this problem in an entirely different manner, by using an evolutionary, population-based approach, and directly searching the policy space to avoid carrying out an optimization over the entire action space. Such an approach has proven effective in attacking many traditional optimization problems, both deterministic (combinatorial) and stochastic. Our work applies this approach to infinite horizon discounted cost MDP models and results in a novel algorithm we call Evolutionary Random Policy Search (ERPS).

For a given MDP problem, ERPS proceeds iteratively by constructing and solving a sequence of sub-MDP problems, which are MDPs defined on smaller policy spaces. At each iteration of the algorithm, two steps are fundamental: (1) The sub-MDP problem constructed in the previous iteration is approximately solved by using a variant of the policy improvement technique called policy improvement with cost swapping (PICS), and a policy called an elite policy is generated. (2) Based on the elite policy, a group of policies is then obtained by using a "nearest neighbor" heuristic and random sampling of the entire action space, from which a new sub-MDP is created by restricting the original MDP problem (e.g., cost structure, transition probabilities) on the current available subsets of actions. Under some appropriate assumptions, we show that the sequence of elite policies converges with probability one to an optimal policy.

We briefly review the relatively sparse research literature applying evolutionary search methods such as genetic algorithms (GAs) and simulated annealing algorithms (SAs) for solving MDPs. Wells et al. (1999) use GAs to find good finite horizon policies for partially observable MDPs and discuss the effects of different GA parameters. Barash (1999) proposes a genetic search in policy space for solving infinite horizon discounted MDPs, and by comparing with the standard policy iteration (PI), concludes that it is unlikely that policy search based on GAs can offer a competitive approach in cases where PI is implementable. More recently, Chang et al. (2002) propose an algorithm called evolutionary policy iteration (EPI) for solving infinite horizon discrete MDPs with large action spaces by imitating the standard procedures of GAs. Although the algorithm is guaranteed to converge with probability one, no performance comparisons with existing techniques are provided, and the theoretical convergence requires the action space to be finite.

ERPS shares some similarities with the EPI algorithm introduced in Chang et al. (2002), where a sequence of "elite" policies is also produced at successive iterations of the algorithm.

However, the fundamental differences are that in EPI, policies are treated as the most essential elements in optimization, and each "elite" policy is directly generated from a group of policies. On the other hand, in our approach, policies are regarded as intermediate constructions from which sub-MDP problems are then constructed and solved; EPI follows the general framework of GAs, and thus operates only at the global level, which usually results in slow convergence. In contrast, ERPS combines global search with a local enhancement step (the "nearest neighbor" heuristic) that leads to rapid convergence once a policy is found in a small neighborhood of an optimal policy. We argue that our approach substantially improves the performance of the EPI algorithm while maintaining the computational complexity at relatively the same level.

Perhaps the most straightforward and the most commonly used numerical approach in dealing with MDPs with uncountable action spaces is via the use of discretization (cf. Rust 1997). In practice, this could lead to computational difficulties, either resulting in an action space that is too large or in a solution that is not accurate enough. In contrast, our approach works directly on the action space, requiring no explicit discretization, and the adaptive version of the algorithm we proposed improves the efficiency of the search process and produces high quality solutions. As in standard approaches such as PI and VI, the computational complexity of each iteration of ERPS is polynomial in the size of the state space, but unlike these procedures, it is insensitive to the size of the action space, making the algorithm a promising candidate for problems with relatively small state spaces but uncountable action spaces.

The rest of the paper is organized as follows. In Section 2, we begin with the problem setting. In Sections 3 and 4, we give a detailed description of the ERPS algorithm and present some convergence results. In Section 5, we introduce an adaptive version of ERPS. Numerical studies and comparisons with EPI and PI are reported in Section 6. Finally some future research topics are outlined in Section 7.

2 Problem Setting

We consider the infinite horizon discounted cost MDP problem $G = (X, A, P, R, \alpha)$ with finite state space X, a general action space A, a bounded non-negative cost function $R : X \times A \to \Re^{+} \cup \{0\}$, a fixed discount factor $\alpha \in (0, 1)$, and a transition function P that maps a state-action pair to a probability distribution over X. The probability of transitioning to state $y \in X$ given that we are in state x taking action $a \in A$, is denoted by $P_{x,y}(a)$.

Throughout this paper, unless otherwise specified, we will always denote the set of all stationary deterministic policies $\pi : X \to A$ by Π and the size of the state space by $|X|$. And without lost of generality, we assume that all actions $a \in A$ are admissible for all states $x \in X$.

Define the optimal value associated with an initial state x as

$$\begin{array}{rcl} J^*(x) & = & \inf_{\pi\in\Pi} J^\pi(x), \text{ where} \\ J^\pi(x) & = & E\left[\sum_{t=0}^{\infty} \alpha^t R(x_t, \pi(x_t)) | x_0 = x\right], \ x \in X, \ \alpha \in (0,1), \end{array} \tag{1}$$

x_t is the state of the MDP at time t, and $E(\cdot)$ is understood with respect to the sample space induced by the transition probabilities. Assume there exists a stationary optimal policy $\pi^* \in \Pi$ that achieves the optimal value $J^*(x)$ for all initial states $x \in X$. The problem is to find such a policy π^*.

3 Evolutionary Random Policy Search

A high-level description of the ERPS algorithm is summarized in Figure 1. We will provide a detailed discussion in the following subsections.

3.1 Initialization

We start by specifying an *action selection distribution* $\mathcal{P}$, the exploitation probability $q_0 \in [0,1]$, the population size n, and a search range r_i for each state $x_i \in X$. Once chosen, these parameters are fixed throughout the algorithm. We then select an initial group of policies; however, because of the exploration step used in ERPS, the performance of the algorithm is relatively insensitive to this choice. One simple method is to choose each individual policy uniformly from the policy space Π.

The *action selection distribution* $\mathcal{P}$ is a probability distribution over the action space, and will be used to generate sub-MDPs (cf. Section 3.3). Note that $\mathcal{P}$ could be state dependent in general, i.e., we could prescribe for each state $x \in X$ a different action selection distribution according to some prior knowledge of the problem structure. One simple choice of $\mathcal{P}$ is the uniform distribution. The exploitation probability q_0 and the search range r_i will be used to construct sub-MDPs; the detailed discussion of these two parameters is deferred to Section 3.3.

Evolutionary Random Policy Search (ERPS)

- **Initialization:** Specify an *action selection distribution* $\mathcal{P}$, the population size $n > 1$, and the exploitation probability $q_0 \in [0,1]$. Specify a search range r_i for each state $x_i \in X,\ i = 1, \ldots, |X|$. Select an initial population of policies $\Lambda_0 = \{\pi_1^0, \pi_2^0, \ldots, \pi_n^0\}$. Construct the initial sub-MDP as $\mathcal{G}_{\Lambda_0} := (X, \Gamma_0, P, R, \alpha)$, where $\Gamma_0 = \bigcup_x \Lambda_0(x)$. Set $\pi_*^{-1} := \pi_1^0$, $k = 0$.

- **Repeat until a specified stopping rule is satisfied:**

  - **Policy Improvement with Cost Swapping (PICS):**

    - Obtain the value function $J^{\pi_j^k}$ for each $\pi_j^k \in \Lambda_k$.
    - Generate the elite policy for $\mathcal{G}_{\Lambda_k}$ as

$$\pi_*^k(x) = \arg\min_{u \in \Lambda_k(x)} \left\{ R(x,u) + \alpha \sum_y P_{x,y}(u)[\min_{\pi_j^k \in \Lambda_k} J^{\pi_j^k}(y)] \right\},\ \forall x \in X.$$

  - **Sub-MDP Generation:**

```
* for $j = 2$ to $n$
    for $i = 1$ to $|X|$
        generate a random sample $u$ from the uniform distribution over $[0,1]$,
        if $u \le q_0$ (exploitation)
            choose the action $\pi_j^{k+1}(x_i)$ in the neighborhood of $\pi_*^k(x_i)$
            by using the "nearest neighbor" heuristic.
        elseif $u > q_0$ (exploration)
            choose the action $\pi_j^{k+1}(x_i) \in A$ according to $\mathcal{P}$.
        end if
    end for
  end for
```

    - Set the next population generation as $\Lambda_{k+1} = \{\pi_*^k, \pi_2^{k+1}, \ldots, \pi_n^{k+1}\}$.
    - Construct a new sub-MDP problem as $\mathcal{G}_{\Lambda_{k+1}} := (X, \Gamma_{k+1}, P, R, \alpha)$, where $\Gamma_{k+1} = \bigcup_x \Lambda_{k+1}(x)$.
    - $k \leftarrow k + 1$.

Figure 1: Evolutionary Random Policy Search

3.2 Policy Improvement with Cost Swapping

As mentioned earlier, the idea behind ERPS is to randomly split a large MDP problem into a sequence of smaller, manageable MDPs, and to extract a possibly convergent sequence of policies via solving these smaller problems. For a given policy population $\Lambda = \{\pi_1, \pi_2, \ldots, \pi_n\}$, if we restrict the original MDP (e.g., costs, transition probabilities) on the subsets of actions $\Lambda(x) := \{\pi_1(x), \pi_2(x), \ldots, \pi_n(x)\}\ \forall x \in X$, then a sub-MDP problem is induced from Λ as $\mathcal{G}_\Lambda := (X, \Gamma, P, R, \alpha)$, where $\Gamma := \bigcup_x \Lambda(x)$. Note that in general $\Lambda(x)$ is a multi-set, which means that the set may contain repeated elements; however, we can always discard the redundant members and view $\Lambda(x)$ as the set of admissible actions at state x. Since ERPS is an iterative random search algorithm, rather than attempting to solve $\mathcal{G}_\Lambda$ exactly, it is more efficient to utilize some approximation schemes and to obtain an improved policy and/or good candidate policies with some worst-case performance guarantee.

Here we adopt a variant of the policy improvement technique to find an "elite" policy, one that is superior to all of the policies in the current population, by executing the following two steps:

Step 1: Obtain the value functions $J^{\pi_j},\ j = 1, \ldots, n$, by solving the equations:

$$J^{\pi_j}(x) = R(x, \pi_j(x)) + \alpha \sum_y P_{x,y}(\pi_j(x)) J^{\pi_j}(y),\ \forall x \in X. \tag{2}$$

Step 2: Compute the elite policy π_* by

$$\pi_*(x) = \operatorname*{arg\,min}_{u \in \Lambda(x)} \left\{ R(x, u) + \alpha \sum_y P_{x,y}(u) [\min_{\pi_j \in \Lambda} J^{\pi_j}(y)] \right\},\ \forall x \in X. \tag{3}$$

Since in equation (3), we are basically performing the policy improvement on the "swapped cost" $\min_{\pi_j \in \Lambda} J^{\pi_j}(x)$, we call this procedure "policy improvement with cost swapping" (PICS). We remark that the "swapped cost" $\min_{\pi_j \in \Lambda} J^{\pi_j}(x)$ may not be the value function corresponding to any policy. We now show that the elite policy generated by PICS improves any policy in Λ.

Theorem 1 *Given $\Lambda = \{\pi_1, \pi_2, \ldots, \pi_n\}$, let $\bar{J}(x) = \min_{\pi_j \in \Lambda} J^{\pi_j}(x)\ \forall x \in X$, and let*

$$\mu(x) = \operatorname*{arg\,min}_{u \in \Lambda(x)} \left\{ R(x, u) + \alpha \sum_y P_{x,y}(u(x)) \bar{J}(y) \right\}.$$

Then $J^\mu(x) \le \bar{J}(x),\ \forall x \in X$. Furthermore, if μ is not optimal for $\mathcal{G}_\Lambda$, then $J^\mu(x) < \bar{J}(x)$ for at least one $x \in X$.

Proof: Let $J_0(x) = R(x, \mu(x)) + \alpha \sum_y P_{x,y}(\mu(x)) \bar{J}(y)$, and consider the sequence $J_1(x), J_2(x), \ldots$ generated by the recursion $J_{i+1}(x) = R(x, \mu(x)) + \alpha \sum_y P_{x,y}(\mu(x)) J_i(y)$, $\forall i = 0, 1, 2, \ldots$. At state x, by definition of $\bar{J}(x)$, there exists π_j such that $\bar{J}(x) = J^{\pi_j}(x)$. It follows that

$$\begin{aligned} J_0(x) &\leq R(x, \pi_j(x)) + \alpha \textstyle\sum_y P_{x,y}(\pi_j(x))\bar{J}(y) \\ &\leq R(x, \pi_j(x)) + \alpha \textstyle\sum_y P_{x,y}(\pi_j(x))J^{\pi_j}(y) \\ &= J^{\pi_j}(x) \\ &= \bar{J}(x) \ , \end{aligned}$$

and since x is arbitrary, we have

$$\begin{aligned} J_1(x) &= R(x, \mu(x)) + \alpha \textstyle\sum_y P_{x,y}(\mu(x))J_0(y) \\ &\leq R(x, \mu(x)) + \alpha \textstyle\sum_y P_{x,y}(\mu(x))\bar{J}(y) \\ &= J_0(x) \ . \end{aligned}$$

By induction $J_{i+1}(x) \leq J_i(x)$, $\forall\, x \in X$ and $\forall\, i = 0, 1, 2, \ldots$. It is well known (Bertsekas 1995) that the sequence $J_0(x), J_1(x), J_2(x), \ldots$ generated above converges to $J^{\mu}(x)$, $\forall\, x \in X$. Therefore $J^{\mu}(x) \leq \bar{J}(x)$, $\forall\, x$. If $J^{\mu}(x) = \bar{J}(x)$, $\forall\, x \in X$, then PICS reduces to the standard policy improvement on policy μ, and it follows that μ satisfies the Bellman's optimality equation and is thus optimal for $\mathcal{G}_{\Lambda}$. Hence we must have $J^{\mu}(x) < \bar{J}(x)$ for some $x \in X$ whenever μ is not optimal. ∎

Now at the *kth* iteration, given the current policy population Λ_k, we compute the *kth* elite policy π_*^k via PICS. According to Theorem 1, the elite policy improves any policy in Λ_k, and since π_*^k is directly used to generate the $(k+1)th$ sub-MDP (cf. Figure 1 and Section 3.3), the following monotonicity property is immediately clear:

Corollary 1 *For all $k \geq 0$,*

$$J^{\pi_*^{k+1}}(x) \leq J^{\pi_*^k}(x), \quad \forall\, x \in X.$$

Proof: Follows by induction. ∎

In EPI, an "elite" policy is also obtained at each iteration by a method called "policy switching". Unlike PICS, policy switching constructs an elite policy by directly manipulating each individual policy in the population. To be precise, for the given policy population $\Lambda = \{\pi_1, \pi_2, \ldots, \pi_n\}$, the elite policy is constructed as

$$\pi_*(x) \in \left\{ \underset{\pi_i \in \Lambda}{\arg\min} (J^{\pi_i}(x))(x) \right\}, \ \forall\, x \in X, \tag{4}$$

where the value functions J^{π_i}, $\forall\, \pi_i \in \Lambda$ are obtained by solving equation (2). It is proven in Chang et al. (2002) that the elite policy π_* generated by policy switching also improves any policy in the population Λ. Note that the computational complexity of executing equation (4) is $O(n|X|)$.

We now provide a heuristic comparison between PICS and policy switching. For a given group of policies Λ, let Ω be the policy space for the sub-MDP $\mathcal{G}_\Lambda$; it is clear that the size of Ω is on the order of $n^{|X|}$. Policy switching only takes into account each individual policy in Λ, while PICS tends to search the entire space Ω, which is a much larger set than Λ. Although it is not clear in general that the elite policy generated by PICS improves the elite policy generated by policy switching, since the policy improvement step is quite fast and it focuses on the best policy updating directions, we believe this will be the case in many situations. For example, consider the case where one particular policy, say $\bar{\pi}$, dominates all other policies in Λ. It is obvious that policy switching will choose $\bar{\pi}$ as the elite policy; thus no further improvement can be achieved. In contrast, PICS considers the sub-MDP $\mathcal{G}_\Lambda$; as long as $\bar{\pi}$ is not optimal for $\mathcal{G}_\Lambda$, a better policy can always be obtained.

The computational complexity of each iteration of PICS is approximately the same as that of policy switching, because step 1 of PICS, i.e., equation (2), which is also used by policy switching, requires solution of n systems of linear equations, and the number of operations required by using a direct method (e.g., Gaussian Elimination) is $O(n|X|^3)$, and this dominates the cost of step 2, which is at most $O(n|X|^2)$.

3.3 Sub-MDP Generation

The description of the "sub-MDP generation" step in Figure 1 is only at a conceptual level. In order to elaborate, we need to distinguish between two cases. We first consider the discrete action space case; then we discuss the setting where the action space is continuous.

3.3.1 Discrete Action Spaces

According to Corollary 1, the performance of the elite policy at the current iteration is no worse than the performances of the elite policies generated at previous iterations. Our concern now is how to achieve continuous improvements among the elite policies found at consecutive iterations. One possibility is to use unbiased random sampling and choose at each iteration a sub-MDP problem by making use of the *action selection distribution* $\mathcal{P}$. The sub-MDPs at successive iterations are then independent of one another, and it is intuitively clear that we may obtain improved elite policies after a sufficient number of iterations. Such an unbiased sampling scheme is very effective in escaping local optima and is often useful in finding a good candidate solution. However, in practice persistent improvements will be more and more difficult to achieve as the number of iterations (sampling instances) increases, since the probability of finding better elite policies becomes smaller and smaller. See Lourenco, Martin, and Stützle (2001) for a more insightful discussion in a global optimization context. Thus, it appears that a biased sampling scheme could be more helpful, which can be accomplished by using a "nearest neighbor" heuristic.

To achieve a biased sampling configuration, ERPS combines exploitation ("nearest neighbor" heuristic) with exploration (unbiased sampling). The key to balance these two types of searches is the use of the exploitation probability q_0. For a given elite policy π, we construct a new policy, say $\hat{\pi}$, in the next population generation as follows: At each state $x \in X$, with probability q_0, $\hat{\pi}(x)$ is selected from a small neighborhood of $\pi(x)$; and with probability $1 - q_0$, $\hat{\pi}(x)$ is chosen by using the unbiased random sampling. The preceding procedure is performed repeatedly until we have obtained $n - 1$ new policies, and the next population generation is simply formed by the elite policy π and the $n - 1$ newly generated policies. Intuitively, on the one hand, the use of exploitation will introduce more robustness into the algorithm and helps to locate the exact optimal policy, while on the other hand, the exploration step will help the algorithm to escape local optima and to find attractive policies quickly. In effect, we see that this idea is equivalent to altering the underlying *action selection distribution*, in that $\mathcal{P}$ is artificially made more peaked around the action $\pi(x)$.

If we assume that A is a non-empty metric space with a defined metric $d(\cdot, \cdot)$, then the "nearest neighbor" heuristic in Figure 1 could be implemented as follows:

Let r_i, a positive integer, be the search range for state x_i, $i = 1, 2, \ldots, |X|$. We assume that $r_i < |A|$ for all i, where $|A|$ is the size of the action space.

- Generate a random variable $l \sim DU(1, r_i)$, where $DU(1, r_i)$ represents the discrete uniform distribution between 1 and r_i. Set $\pi_j^{k+1}(x_i) = a \in A$ such that a is the lth closest action to $\pi_*^k(x_i)$ (measured by $d(\cdot, \cdot)$).

Remark 1: Sometimes the above procedure is not easy to implement. It is often necessary to index a possibly high-dimensional metric space, whose complexity will depend on the dimension of the problem and the cost in evaluating the distance functions. However, we note that the action spaces of many MDP problems are subsets of $\Re^N$, where a lot of efficient methods can be applied, such as Kd-trees (Bentley 1979) and R-trees (Guttman 1984). The most favorable situation is an action space that is "naturally ordered", e.g., in inventory control problems where actions are the number of items to be ordered $A = \{0, 1, 2, \cdots\}$, in which case the indexing and ordering becomes trivial.

Remark 2: In EPI, policies in a new generation are generated by the so-called "policy mutation" procedure, where two types of mutations are considered: "global mutation" and "local mutation". The algorithm first decides whether to mutate a given policy π "globally" or "locally" according to a mutation probability P_m. Then at each state x, $\pi(x)$ is mutated with probability P_g or P_l, where P_g and P_l are the respective predefined global mutation and local mutation probabilities. It is assumed that $P_g \gg P_l$; the idea is that "global mutation" helps the algorithm to get out of local optima and "local mutation" helps the algorithm to fine-tune the solution. If a mutation is to occur, the action is changed by using the *action selection probability* $\mathcal{P}$. As a result, we see that each action in a new policy generated by "policy mutation" either remains unchanged or is altered by pure random sampling; although the so-called "local mutation" is used, no local search element is actually involved in the process. Thus, as mentioned before, the algorithm only operates at the global level. We note that this is essentially equivalent to setting the exploitation probability $q_0 = 0$ in our approach.

3.3.2 Continuous Action Spaces

The biased sampling idea in the previous section can be naturally extended to MDPs with continuous action spaces. We let $\mathcal{B}_A$ be the smallest σ-algebra containing all the open sets in A, and choose the *action selection distribution* $\mathcal{P}$ as a probability measure defined on $(A, \mathcal{B}_A)$. Again, denote the metric defined on A by $d(\cdot,\cdot)$.

By following the "nearest neighbor" heuristic, we now give a general implementation of the exploitation step in Figure 1.

Let $r_i > 0$ denote the search range for state x_i, $i = 1, 2, \ldots, |X|$.

- Choose an action uniformly from the set of neighbors $\{a : d(a, \pi_*^k(x_i)) \leq r_i,\ a \in A\}$.

Note the difference in the search range r_i between the discrete action space case and the continuous action space case. In the former case, r_i is a positive integer indicating the number of candidate actions that are the closest to the current elite action $\pi_*^k(x_i)$, whereas in the latter case, r_i is the distance from the current elite action, which may take any positive value.

If we further assume that A is a non-empty open connected subset of $\Re^N$ with some metric (e.g., the infinity-norm), then a detailed implementation of the above exploitation step is as follows.

- Generate a random vector $\lambda^i = (\lambda_1^i, \ldots, \lambda_N^i)^T$ with each $\lambda_h^i \sim U[-1, 1]$ independent for all $h = 1, 2, \ldots, N$, and choose the action $\pi_j^{k+1}(x_i) = \pi_*^k(x_i) + \lambda^i r_i$.

- *If* $\pi_j^{k+1}(x_i) \notin A$, then repeat the above step.

In this specific implementation, the same search range r_i is used along all directions of the action space. However, in practice, it may often be useful to generalize r_i to a N-dimensional vector, where each component controls the search range in a particular direction of the action space.

Remark 3: Note that the action space does not need to have any structure other than being a metric space. The metric $d(\cdot,\cdot)$ used in the "nearest neighbor" heuristic implicitly imposes a structure on the action space. It follows that the efficiency of the algorithm depends on how the metric is actually defined. Like most of the random search methods for global optimizations, our approach is designed to explore the structure that good policies tend to be clustered together. Thus, in our context, a good metric should have a good potential in representing this structure. For example, the discrete metric (i.e., $d(a,a) = 0 \; \forall \; a \in A$ and $d(a,b) = 1 \; \forall \; a, b \in A, \;\; a \neq b$) should never be considered as a good choice, since it does not provide us with any useful information about the action space. For a given action space, a good metric always exists but may not be known a priori. In the special case where the action space is a subset of $\Re^N$, we take the Euclidean metric as the default metric, this is in accord with most of the optimization techniques employed in $\Re^N$.

3.4 Stopping Rule

Different stopping criteria can be used. The simplest one is to stop the algorithm when a predefined maximum number of iterations is reached. In the numerical experiments reported in Section 6, we use one of the most common stopping rules in standard GAs (Srinivas and Patnaik 1994; Wells et al. 1999; Chang et al. 2002): the algorithm is stopped when no further improvement in the value function is obtained for several, say K, consecutive iterations. To be precise, we stop the algorithm if $\exists \; k > 0, \;$ such that $\|J^{\pi_*^{k+m}} - J^{\pi_*^k}\| = 0 \; \forall \; m = 1, 2, \ldots, K$.

4 Convergence of ERPS

In this section, we discuss the convergence properties of ERPS. To do so, the following notation is necessary.

As before, denote by $d(\cdot,\cdot)$ the metric on the action space A. We define the distance between two policies π^1 and π^2 by

$$d_\infty(\pi^1, \pi^2) := \max_{1 \le i \le |X|} d(\pi^1(x_i), \pi^2(x_i)).$$

For a given policy $\hat{\pi} \in \Pi$ and any $\sigma > 0$, we further define the σ-neighborhood of $\hat{\pi}$ by

$$\mathcal{N}(\hat{\pi}, \sigma) := \{\pi |\ d_\infty(\hat{\pi}, \pi) \leq \sigma,\ \forall \pi \in \Pi\}.$$

For each policy $\pi \in \Pi$, we also define P_π as the transition matrix whose $(x, y)th$ entry is $P_{x,y}(\pi(x))$ and R_π as the one-stage cost vector whose $(x)th$ entry is $R(x, \pi(x))$.

As the ERPS method is randomized, different runs of the algorithm will give different sequences of elite policies (i.e., sample paths); thus the algorithm induces a probability distribution over the set of all sequences of elite policies. We denote by $\hat{\mathcal{P}}(\cdot)$ and $\hat{E}(\cdot)$ the probability and expectation taken with respect to this distribution.

Let $\|\cdot\|_\infty$ denote the infinity-norm, given by $\|J\|_\infty := \max_{x \in X} |J(x)|$. We have the following convergence result for the ERPS algorithm.

Theorem 2 *Let π^* be an optimal policy with corresponding value function J^{π^*}, and let the sequence of elite policies generated by ERPS together with their corresponding value functions be denoted by $\{\pi_*^k,\ k = 1, 2, \ldots\}$ and $\{J^{\pi_*^k},\ k = 1, 2, \ldots\}$, respectively. Assume that:*

1. *$q_0 < 1$.*

2. *For any given $\ell > 0$, $\mathcal{P}(\{a|\ d(a, \pi^*(x)) \leq \ell,\ a \in A\}) > 0,\ \forall x \in X$, (recall that $\mathcal{P}(\cdot)$ is a probability measure on the action space A).*

3. *There exist constants $\sigma > 0$, $\phi > 0$, $L_1 < \infty$, and $L_2 < \infty$, such that for all $\pi \in \mathcal{N}(\pi^*, \sigma)$ we have $\|P_\pi - P_{\pi^*}\|_\infty \leq \min\left\{L_1 d_\infty(\pi, \pi^*), \frac{1-\alpha}{\alpha} - \phi\right\}$ $(0 < \alpha < 1)$, and $\|R_\pi - R_{\pi^*}\|_\infty \leq L_2 d_\infty(\pi, \pi^*)$.*

Then for any given $\varepsilon > 0$, there exists a random variable $\mathcal{M}_\varepsilon > 0$ with $\hat{E}(\mathcal{M}_\varepsilon) < \infty$ such that $\|J^{\pi_*^k} - J^{\pi^*}\|_\infty \leq \varepsilon\ \ \forall\, k \geq \mathcal{M}_\varepsilon$.

Remark 4: Assumption 1 restricts the exploitation probability from pure local search. Assumption 2 simply requires that any "ball" that contains the optimal policy will have a strictly positive probability measure. It is trivially satisfied if the set $\{a|d(a, \pi^*(x)) \leq \ell,\ a \in A\}$ has a positive (Borel) measure $\forall\ x \in X$ and the *action selection distribution* $\mathcal{P}$ has infinite tails (e.g., Gaussian, exponential). Assumption 3 imposes some Lipschitz type of conditions on P_π and R_π; as we will see, it formalizes the notion that near optimal policies are clustered together (cf. remark 3). The assumption can be verified if P_π and R_π are explicit functions of π (which is the case in our numerical examples; cf. Section 6). For a given $\varepsilon > 0$, such a policy π satisfying $\|J^\pi - J^{\pi^*}\|_\infty \leq \varepsilon$ is referred to as an ε-optimal policy (cf. Bertsekas 1995).

Remark 5: The result in Theorem 2 implies the a.s. convergence of the sequence $\{J^{\pi_*^k}, k = 0, 1, \ldots\}$ to the optimal value function J^{π^*}. To see this, note that Theorem 2 implies that $\hat{\mathcal{P}}(\|J^{\pi_*^k} - J^{\pi^*}\|_\infty > \varepsilon) \to 0$ as $k \to \infty$ for every given ε, which means that the sequence converges in probability. Furthermore, since $\|J^{\pi_*^k} - J^{\pi^*}\|_\infty \leq \varepsilon \ \ \forall\, k \geq \mathcal{M}_\varepsilon$ is equivalent to $\sup_{\bar{k} \geq k} \|J^{\pi_*^{\bar{k}}} - J^{\pi^*}\|_\infty \leq \varepsilon \ \ \forall\, k \geq \mathcal{M}_\varepsilon$, we will also have $\hat{\mathcal{P}}(\sup_{\bar{k} \geq k} \|J^{\pi_*^{\bar{k}}} - J^{\pi^*}\|_\infty > \varepsilon) \to 0$ as $k \to \infty$, and the a.s. convergence thus follows.

Proof: The first step in the proof is to derive an upperbound for $\|J^\pi - J^{\pi^*}\|_\infty$ in terms of the distance $d_\infty(\pi, \pi^*)$. For policy π^* and policy π we have:

$$\begin{aligned} J^{\pi^*} &= R_{\pi^*} + \alpha P_{\pi^*} J^{\pi^*}, && (5) \\ J^{\pi} &= R_{\pi} + \alpha P_{\pi} J^{\pi}. && (6) \end{aligned}$$

Now subtract the above two equations and define $\Delta J^{\pi^*} = J^\pi - J^{\pi^*}$, $\Delta P_{\pi^*} = P_\pi - P_{\pi^*}$ and $\Delta R_{\pi^*} = R_\pi - R_{\pi^*}$. We have

$$\Delta J^{\pi^*} = [I - (I - \alpha P_{\pi^*})^{-1} \alpha \Delta P_{\pi^*}]^{-1} (I - \alpha P_{\pi^*})^{-1} (\alpha \Delta P_{\pi^*} J^{\pi^*} + \Delta R_{\pi^*}). \tag{7}$$

Taking the norm of both sides of (7) and using the consistency property of the operator norm (i.e., $\|AB\| \leq \|A\| \cdot \|B\|$), it follows that

$$\|\Delta J^{\pi^*}\|_\infty \leq \|[I - (I - \alpha P_{\pi^*})^{-1} \alpha \Delta P_{\pi^*}]^{-1}\|_\infty \|(I - \alpha P_{\pi^*})^{-1}\|_\infty (\alpha \|\Delta P_{\pi^*}\|_\infty \|J^{\pi^*}\|_\infty + \|\Delta R_{\pi^*}\|_\infty). \tag{8}$$

By assumption 3, we have $\|\Delta P_{\pi^*}\|_\infty < \frac{1-\alpha}{\alpha}$, thus

$$\begin{aligned} \|(I - \alpha P_{\pi^*})^{-1} \alpha \Delta P_{\pi^*}\|_\infty &\leq \|(I - \alpha P_{\pi^*})^{-1}\|_\infty \alpha \|\Delta P_{\pi^*}\|_\infty \\ &< \|(I - \alpha P_{\pi^*})^{-1}\|_\infty (1 - \alpha) \\ &< 1. \end{aligned}$$

We now try to divide both sides of equation (8) by $\|J^{\pi^*}\|_\infty$. Before we proceed, we need to distinguish between two cases, $\|J^{\pi^*}\|_\infty = 0$ and $\|J^{\pi^*}\|_\infty \neq 0$.

Case 1. If $R_{\pi^*} = 0$ (i.e., $R(x, \pi^*(x)) = 0$ for all $x \in X$), then we have $J^{\pi^*} = 0$. Thus $\Delta J^{\pi^*} = J^\pi$ and $\Delta R_{\pi^*} = R_\pi$. By noting $\|P_\pi\|_\infty = 1$, it follows from (6) that

$$\|\Delta J^{\pi^*}\|_\infty = \|J^\pi\|_\infty \leq \frac{1}{1 - \alpha \|P_\pi\|_\infty} \|R_\pi\|_\infty = \frac{1}{1 - \alpha} \|\Delta R_{\pi^*}\|_\infty.$$

Then by assumption 3,

$$\|\Delta J^{\pi^*}\|_\infty \leq \frac{L_2}{1 - \alpha} d_\infty(\pi, \pi^*). \tag{9}$$

Case 2. If $R_{\pi^*} > 0$ (i.e., $R(x, \pi^*(x)) > 0$ for some $x \in X$), then from (5), $J^{\pi^*} > 0$. Divide both sides of (8) by $\|J^{\pi^*}\|_\infty$, use the relation that $\|(I-B)^{-1}\| \le \frac{1}{1-\|B\|}$ whenever $\|B\| < 1$ and the consistency property; it immediately follows that

$$\begin{aligned}
\frac{\|\Delta J^{\pi^*}\|_\infty}{\|J^{\pi^*}\|_\infty} &\le \frac{\|(I-\alpha P_{\pi^*})^{-1}\|_\infty}{1-\|(I-\alpha P_{\pi^*})^{-1}\|_\infty \alpha \|\Delta P_{\pi^*}\|_\infty} \left\{ \alpha \|\Delta P_{\pi^*}\|_\infty + \frac{\|\Delta R_{\pi^*}\|_\infty}{\|J^{\pi^*}\|_\infty} \right\} \\
&= \frac{\|(I-\alpha P_{\pi^*})^{-1}\|_\infty \|I-\alpha P_{\pi^*}\|_\infty}{1-\|(I-\alpha P_{\pi^*})^{-1}\|_\infty \alpha \|\Delta P_{\pi^*}\|_\infty} \left\{ \frac{\alpha \|\Delta P_{\pi^*}\|_\infty}{\|I-\alpha P_{\pi^*}\|_\infty} + \frac{\|\Delta R_{\pi^*}\|_\infty}{\|I-\alpha P_{\pi^*}\|_\infty \|J^{\pi^*}\|_\infty} \right\} \\
&\le \frac{\mathcal{K}}{1-\mathcal{K}\frac{\alpha\|\Delta P_{\pi^*}\|_\infty}{\|I-\alpha P_{\pi^*}\|_\infty}} \left\{ \frac{\alpha \|\Delta P_{\pi^*}\|_\infty}{\|I-\alpha P_{\pi^*}\|_\infty} + \frac{\|\Delta R_{\pi^*}\|_\infty}{\|R_{\pi^*}\|_\infty} \right\} \\
&\le \frac{\mathcal{K}}{1-\mathcal{K}\frac{\alpha\|\Delta P_{\pi^*}\|_\infty}{\|I-\alpha P_{\pi^*}\|_\infty}} \left\{ \frac{\alpha L_1}{\|I-\alpha P_{\pi^*}\|_\infty} + \frac{L_2}{\|R_{\pi^*}\|_\infty} \right\} d_\infty(\pi, \pi^*),
\end{aligned} \tag{10}$$

where $\mathcal{K} = \|(I-\alpha P_{\pi^*})^{-1}\|_\infty \|I-\alpha P_{\pi^*}\|_\infty$.

In view of (9) and (10), we conclude that for any given $\varepsilon > 0$, there exists a $\theta > 0$ such that for any $\pi \in \mathcal{N}(\pi^*, \sigma)$ where

$$d_\infty(\pi, \pi^*) := \max_{1 \le i \le |X|} d(\pi(x_i), \pi^*(x_i)) \le \theta,$$

we have $\|J^\pi - J^{\pi^*}\|_\infty = \|\Delta J^{\pi^*}\|_\infty \le \varepsilon$. Note that $\max_{1\le i \le |X|} d(\pi(x_i), \pi^*(x_i)) \le \theta$ is equivalent to

$$d(\pi(x_i), \pi^*(x_i)) \le \theta, \ \forall \ i = 1, 2, \ldots, |X|. \tag{11}$$

By assumption 2, the set of actions that satisfies (11) will have a strictly positive probability measure, and since $q_0 < 1$, it follows that the probability a population generation does not contain a policy in the neighborhood $\mathcal{N}(\pi^*, \min\{\theta, \sigma\})$ of the optimal policy is strictly less than 1. Let ψ be the probability that a randomly constructed policy is in $\mathcal{N}(\pi^*, \min\{\theta, \sigma\})$. Then at each iteration, the probability that at least one policy is obtained in $\mathcal{N}(\pi^*, \min\{\theta, \sigma\})$ is $1-(1-\psi)^{n-1}$, where n is the population size. Thus, the initial part of the proof together with Theorem 1 implies that the value function of the elite policy obtained in the next iteration is ε-optimal at least with probability $1-(1-\psi)^{n-1}$. Let $\mathcal{M}_\varepsilon$ denote the number of iterations required to generate such an elite policy for the first time. By the monotonicity of the sequence $\{J^{\pi^k_*}, \ k = 0, 1, \ldots\}$ (cf. Corollary 1), it is clear that $\|J^{\pi^k_*} - J^{\pi^*}\|_\infty \le \varepsilon \ \forall \ k \ge \mathcal{M}_\varepsilon$. Now consider a random variable $\bar{\mathcal{M}}$ that is geometrically distributed with a success probability of $1-(1-\psi)^{n-1}$. It is not difficult to see that $\bar{\mathcal{M}}$ dominates $\mathcal{M}_\varepsilon$ stochastically (i.e., $\bar{\mathcal{M}} \ge_{st} \mathcal{M}_\varepsilon$), and because $\psi > 0$, it follows that $\hat{E}(\mathcal{M}_\varepsilon) \le \hat{E}(\bar{\mathcal{M}}) = \frac{1}{1-(1-\psi)^{n-1}} < \infty$. ■

Remark 6: In the above proof, we have used the infinity-norm. Since in finite dimensional spaces all norms are equivalent (cf. Demmel 1997), similar results can also be easily established by using different norms, e.g., the Euclidean-norm.

Remark 7: It should be noted that the result presented in Theorem 2 is rather theoretical, because nothing can be said about the convergence rate of the algorithm as well as how much improvement can be achieved at each iteration. As a consequence, the random variable $\mathcal{M}_\varepsilon$ could be extremely large in practice.

Note that for a discrete finite action space, assumption 3 in Theorem 2 is automatically satisfied, and assumption 2 also holds trivially if we take $\mathcal{P}(a) > 0$ for all actions $a \in A$. Furthermore, when the action space is finite, there always exists an $\varepsilon > 0$ such that the only ε-optimal policy is the optimal policy itself. We have the following stronger convergence result for ERPS when the action space is finite.

Corollary 2 *(Discrete finite action space) If the action space is finite, $q_0 < 1$, and the action selection distribution $\mathcal{P}(a) > 0 \; \forall \, a \in A$, then there exists a random variable $\mathcal{M} > 0$ such that $\hat{\mathcal{P}}(\mathcal{M} < \infty) = 1$ and $\hat{E}(\mathcal{M}) < \infty$, and $J^{\pi_*^k} = J^{\pi^*} \; \forall \; k \geq \mathcal{M}$.*

5 Adaptive ERPS

The search range parameter r_i in ERPS is fixed throughout the algorithm. Intuitively, small search ranges concentrate the search in small regions around the desirable points and are helpful in refining promising solutions, but they often lead to small improvements in the cost function, thus slowing down the convergence process. On the other hand, large search ranges typically reduce the number of search steps needed to find a good or near optimal solution, but can be less effective in developing finer details around desirable points and may result in less accurate solutions. In this section, we present a modification of the ERPS method in which the value of the search range parameter may change from one iteration to another. The idea is to adaptively shrink and expand the search range so that we can speed up the convergence process without sacrificing the solution quality. A detailed description of the adaptive ERPS is given in Figure 2, where we only consider the continuous action space case; the discrete action space version can be constructed similarly.

We start by running ERPS with an initially specified search range r (for simplicity, we assume that the same search range is prescribed for all states), and monitor the performance of the elite policy obtained at each iteration. If no improvements among the elite policies

```
Adaptive ERPS
  • Initialization: Specify an initial search range $r$, parameters $K$, $1 < K_1 < K$,
    $K_2 > 1$, $K_3 > 1$, $\gamma > 1$ and a tolerance level $\varepsilon > 0$, where $K$ is the stopping control
    parameter as in ERPS. Set $\imath \leftarrow 0$, $\jmath \leftarrow 0$, and $h \leftarrow 0$.
  • while $(\imath \leq K$ & $h \leq K_3)$
      – Execute ERPS with search range $r$.
      – Search range update:
          if $0 < \|J^{\pi_*^{k+1}} - J^{\pi_*^k}\| \leq \varepsilon$, then set $\imath \leftarrow 0$, $\jmath \leftarrow \jmath + 1$;
          elseif $\|J^{\pi_*^{k+1}} - J^{\pi_*^k}\| = 0$, then set $\imath \leftarrow \imath + 1$, $\jmath \leftarrow 0$;
          else set $\imath \leftarrow 0$, $\jmath \leftarrow 0$.
          end if
          if $\imath \geq K_1$, then set $r_{old} \leftarrow r$, $r \leftarrow r \cdot \frac{1}{\gamma}$. end if
          if $\jmath \geq K_2$, then set $r \leftarrow r \cdot \gamma$. end if
          if $r = r_{old}$, then set $h \leftarrow h + 1$; else set $h \leftarrow 0$. end if
    end while
```

Figure 2: Adaptive ERPS

are achieved for several, say K_1, consecutive iterations, then it indicates that the current search range may be too large, and we decrease it by a factor $\gamma > 1$. On the other hand, if for some consecutive iterations, say K_2, the improvements are non-zero but smaller than some given tolerance ε, then it is likely that the current search range is too small, and we increase it by γ until the improvement is greater than the specified tolerance level. The search range is updated repeatedly until it has been alternating between two values for K_3 times. Intuitively, the adaptive ERPS ensures that each improvement in the elite policy is (approximately) at least ε; when no further improvement is available either by increasing or by decreasing the search range, the value function obtained will be within distance ε of the optimal cost, i.e., the resulting elite policy is approximately ε-optimal.

Note that the validity of the ε-optimality claim relies on the assumption that if there is an improvement of at least ε available, then the algorithm will be able to find it via adaptive adjustment of the search range. The above approach retains the theoretical convergence properties of the original ERPS method and can be applied, at least in principle, to many types of action spaces as long as a metric can be specified; however, we must again emphasize that the efficiency of the approach will depend on the structure of the problem to be solved and how the underlying metric is actually defined.

6 Numerical Examples

In this section, we apply ERPS to two discrete-time controlled queueing problems and compare its performance with that of EPI (Chang et al. 2002) and standard PI. For ERPS, the same search range parameter is prescribed for all states, denoted by a single variable r, and the *action selection distribution* $\mathcal{P}$ is chosen to be the uniform distribution. All computations were performed on an IBM PC with a 2.4 GHz Pentium 4 processor and 512 MB memory, and the computational time units are in seconds.

6.1 A One-Dimensional Queueing Example

The following example is adapted from de Farias and Van Roy 2003 (cf. also Bertsekas 1995). We consider a finite capacity single-server queue with controlled service completion probabilities. Assume that a server can serve only one customer in a period, and the service of a customer begins/ends only at the beginning/end of any period. Customers arrive at a queue independently with probability $p = 0.2$, and there is at most one arrival per period (so no arrival with probability 0.8). The maximum queue length is $\mathcal{L}$, and an arrival that finds $\mathcal{L}$ customers in the queue is lost. We let x_t, the state variable, be the number of customers in the system at the beginning of period t. The action to be chosen at each state is the service completion probability a, which takes value in a set A. In period t, a possible service completion is generated with probability $a(x_t)$, a cost of $R(x_t, a(x_t))$ is incurred, and resulting in a transition to state x_{t+1}. The goal is to choose the optimal service completion probability for each state such that the total discounted cost $E[\sum_{t=0}^{\infty} \alpha^t R(x_t, a(x_t))]$ is minimized.

6.1.1 Discrete Action Space

Two different choices of one-stage cost functions are considered: (i) a simple cost function that is convex in both state and action; (ii) a complicated non-convex cost function. The MDP problem resulting from case (i) may possess some nice properties (e.g., free of multiple local optimal solutions), so finding an optimal solution should be a relatively easy task; whereas the cost function in case (ii) introduces some further computational difficulties (e.g., multiple local minima), intended to more fully test the effectiveness of a global algorithm like

ERPS. For both cases, unless otherwise specified, the following parameter settings are used: maximum queue length $\mathcal{L} = 49$; state space $X = \{0, 1, 2, \ldots, 49\}$; discount factor $\alpha = 0.98$; action set $A = \{10^{-4}k : \; k = 0, 1, \ldots, 10^4\}$; and in ERPS, population size $n = 10$, search range $r = 10$, and the standard Euclidean distance is used to define the neighborhood. All results for ERPS are based on 30 independent replications.

For **case (i)**, the one-stage cost at any period for being in state x and taking action a is given by

$$R(x, a) = x + 50a^2.$$

We test the convergence of ERPS by varying the values of the exploitation probability. Table 1 gives the performances of the algorithm, where the relative error of a value function J is evaluated in the infinity-norm according to the following formula

$$\text{relerr} := \frac{\|J - J^*\|_\infty}{\|J^*\|_\infty}, \tag{12}$$

and J^* is the optimal value function. The computational time required for PI to find the optimal value function J^* was 15 seconds, and the value of $\|J^*\|_\infty$ is approximately 2.32e+03. Test results indicate superior performances of ERPS over PI; in particular, when $q_0 = 0.25, 0.5, 0.75$, ERPS attains the optimal solutions in all 30 independent trials within 2 seconds.

To explore the computational complexity of ERPS, tests were performed on MDPs with increasing numbers of actions; for each problem, the foregoing setting is used except that the action space now takes the form $A = \left\{hk : \; k = 0, 1, \ldots, \frac{1}{h}\right\}$, where h is the mesh size, selected sequentially (one for each problem) from the set $\left\{\frac{1}{100}, \frac{1}{250}, \frac{1}{500}, \frac{1}{1000}, \frac{1}{2500}, \frac{1}{5000}, \frac{1}{10000}, \frac{1}{25000}, \frac{1}{50000}, \frac{1}{100000}, \frac{1}{200000}\right\}$.

In Figure 3, we plot the running time required for PI and ERPS to find the optimal solutions as a function of the number of actions of each MDP considered, where the results for ERPS are the averaged time over 30 independent replications. Empirical results indicate that the computational time for PI increases linearly in the number of actions (note the log-scale used in Figure 3), while the running time required for ERPS does so in an asymptotic sense. We see that ERPS delivers very competitive performances even when the action space is small; when the action space is relatively large (number of actions greater than 10^4), ERPS reduces the computational efforts of PI by roughly a factor of 14. In the experiments, we used a search range $r = 10$ in ERPS, regardless of the size of the action space; we believe the performance of the algorithm could be enhanced by using a search range that is proportional to the size of the action space. Moreover, the computational effort of ERPS can be reduced

q_0	stop rule (K)	Avg. time (std err)	mean relerr (std err)
0.0	2	0.84 (0.03)	7.63e-06 (8.50e-08)
	4	1.41 (0.05)	2.78e-06 (3.29e-07)
	8	2.67 (0.10)	7.83e-07 (1.06e-07)
	16	5.12 (0.16)	1.81e-07 (1.88e-08)
	32	8.91 (0.38)	6.19e-08 (1.07e-08)
0.25	2	0.94 (0.02)	3.32e-09 (1.42e-09)
	4	1.08 (0.02)	9.65e-10 (2.59e-10)
	8	1.24 (0.02)	3.02e-10 (9.51e-11)
	16	1.52 (0.03)	4.54e-11 (3.86e-11)
	32	1.85 (0.04)	0.00e-00 (0.00e-00)
0.50	2	0.92 (0.02)	2.14e-09 (1.29e-09)
	4	1.00 (0.02)	2.53e-10 (1.10e-10)
	8	1.11 (0.02)	7.61e-11 (5.02e-11)
	16	1.27 (0.03)	0.00e-00 (0.00e-00)
0.75	2	1.14 (0.02)	4.14e-10 (2.84e-10)
	4	1.19 (0.02)	2.40e-11 (1.67e-11)
	8	1.27 (0.02)	1.18e-11 (1.18e-11)
	16	1.44 (0.03)	0.00e-00 (0.00e-00)
1.0	2	12.14 (0.02)	1.66e-10 (5.18e-11)
	4	12.19 (0.02)	4.85e-11 (3.49e-11)
	8	12.28 (0.01)	0.00e-00 (0.00e-00)

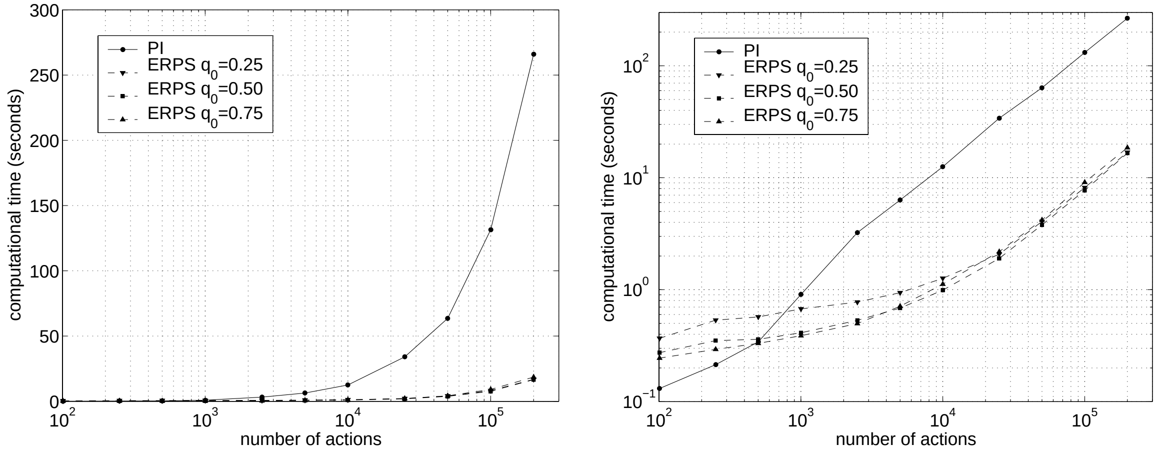
Table 1: Convergence results for ERPS ($n = 10,\ r = 10$) based on 30 independent replications. The standard errors are in parentheses.

considerably if we are merely seeking solutions within some required accuracy rather than insisting on the optimal solution.

For **case (ii)**, we used the following one-stage cost function

$$R(x,a) = x + 5\left[\frac{|X|}{2}sin(2\pi a) - x\right]^2,$$

which induces a tradeoff in choosing between large values of a to reduce the state x and appropriate values of a to make the squared term small. Moreover, since the sine function is not monotone, the resultant MDP problem has a very high number of local minima; some typical locally optimal policies are shown in Figure 4.



(a) (b)

Figure 3: Running time required for PI & ERPS ($n = 10,\ r = 10$, based on 30 independent replications) to find the optimal solutions to MDPs with different numbers of actions, (a) using log-scale for horizontal axis; (b) using log-log plot.

Table 2 shows the convergence properties of EPI and ERPS, where both algorithms start with the same initial population. The computational time required for PI to find the optimal value function J^* was 14 seconds, and the magnitude of $\|J^*\|_\infty$ is approximately 1.03e+05. For EPI, we have tested different sets of parameters (recall from Section 3.3.1, Remark 2, that P_m is the mutation probability; and P_g (P_l) are the predefined global (local) mutation probabilities); the results reported in Table 2 are the best results obtained. Also note that because of the slow convergence of EPI, the values for the stopping control parameter K are chosen much larger than those for ERPS.

In order to demonstrate the role of the exploitation probability q_0 in the ERPS algorithm, we fix the stopping control parameter $K = 10$ and vary q_0. The numerical results are recorded in Table 3, where N$_{opt}$ indicates the number of times an optimal solution was found out of 30 trials. The $q_0 = 1.0$ case corresponds to pure local search. Obviously in this case, the algorithm gets trapped into a local minimum, which has a mean relative error of 5.62e-3. However, note that the standard error is zero, which means that the local minimum is estimated with very high precision. This shows that the "nearest neighbor" heuristic is indeed useful in fine-tuning the solutions. In contrast, the pure random search ($q_0 = 0$) case is helpful in escaping from the local minima, yielding a lower mean relative error of 2.59e-5, but it is not very good in locating the exact optimal solutions, as none was found out of 30 trials. Roughly, increasing q_0 between 0 and 0.5 leads to a more accurate estimation of the optimal solution; however, increasing q_0 on the range 0.6 to 1.0 decreases the quality of the

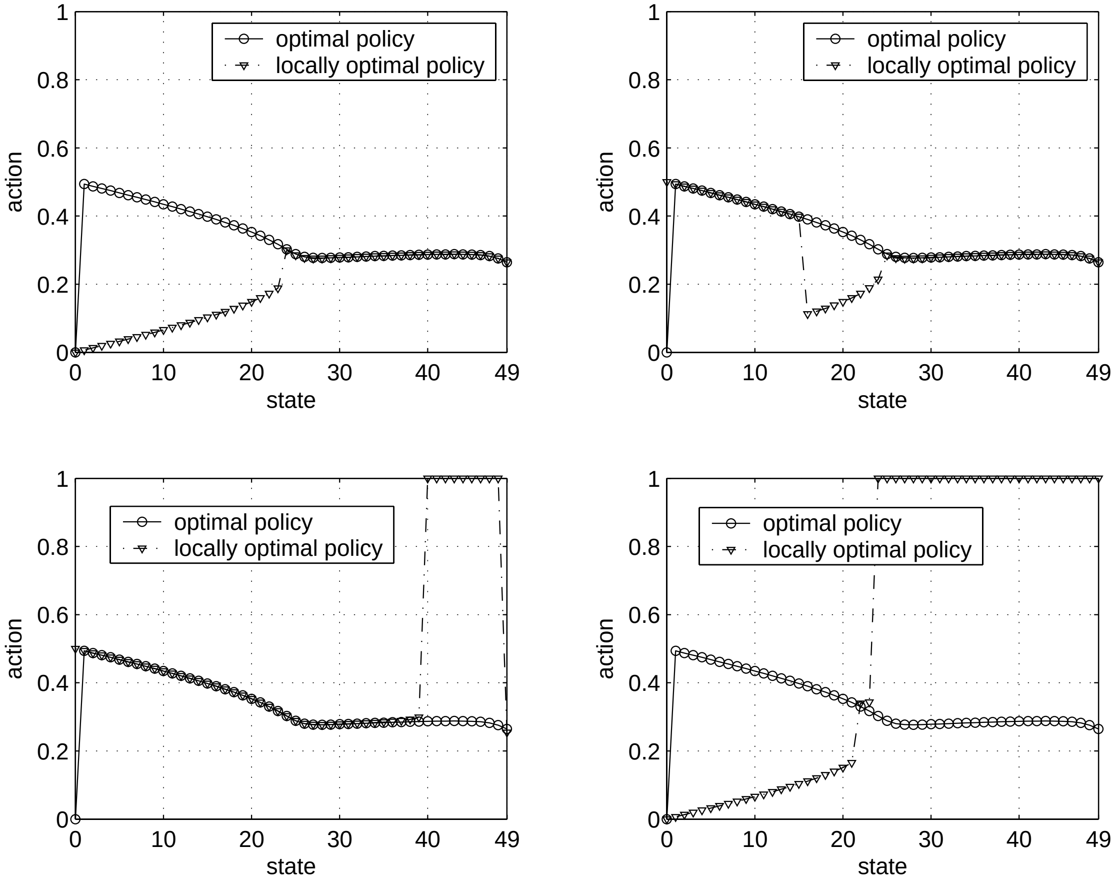


Figure 4: Four typical locally optimal solutions to the test problem.

algorithms	stop rule (K)	Avg. time (std err)	mean relerr (std err)
EPI	20	2.13 (0.11)	1.74e-02 (1.35e-03)
$P_m = 0.1$	40	3.80 (0.16)	1.12e-02 (8.81e-04)
$P_g = 0.9$	80	6.63 (0.34)	7.13e-03 (5.37e-04)
$P_l = 0.1$	160	16.30 (0.59)	3.22e-03 (2.26e-04)
	2	1.03 (0.02)	9.81e-05 (5.17e-05)
ERPS	4	1.12 (0.03)	7.12e-05 (4.95e-05)
$q_0 = 0.5$	8	1.28 (0.03)	2.37e-05 (1.64e-05)
$r = 10$	16	1.50 (0.03)	1.06e-09 (6.59e-10)
	32	1.86 (0.04)	0.00e-00 (0.00e-00)

Table 2: Convergence results for EPI ($n = 10$) & ERPS ($n = 10,\ r = 10$) based on 30 independent replications. The standard errors are in parentheses.

solution, because the local search part begins to gradually dominate, so that the algorithm is more easily trapped in local minima. This also explains why we have larger variances when $q_0 = 0.6, 0.7, 0.8, 0.9$ in Table 3. Notice that the algorithm is very slow in the pure local search case; setting $q_0 < 1$ speeds up the algorithm substantially.

q_0	Avg. time (std err)	N$_{opt}$	mean relerr (std err)
0.0	3.30 (0.13)	0	2.59e-05 (6.19e-06)
0.1	1.96 (0.04)	5	4.51e-08 (8.60e-09)
0.2	1.48 (0.03)	12	1.26e-08 (3.47e-09)
0.3	1.39 (0.02)	24	2.74e-09 (2.02e-09)
0.4	1.28 (0.02)	25	2.69e-05 (1.89e-05)
0.5	1.32 (0.03)	27	8.75e-10 (6.01e-10)
0.6	1.41 (0.04)	25	6.19e-05 (3.20e-05)
0.7	1.50 (0.04)	22	1.53e-04 (6.96e-05)
0.8	1.81 (0.04)	15	3.04e-04 (7.09e-05)
0.9	2.33 (0.08)	11	7.99e-04 (1.63e-04)
1.0	7.86 (0.02)	0	5.62e-03 (0.00e-00)

Table 3: Performance of ERPS with different exploitation probabilities ($n = 10,\ K = 10,\ r = 10$) based on 30 independent replications. The standard errors are in parentheses.

To provide a numerical comparison between the “nearest neighbor” heuristic (biased sampling) and the policy mutation procedure (unbiased sampling), we call the algorithm with the PICS step but policy mutation procedure as algorithm 1. In both ERPS and algorithm 1, we fix the population size $n = 10$, and stop the algorithms only when a desired accuracy is reached. In Table 4, we record the length of time required for different algorithms to reach a relative error of at least 1.0e-6. Indeed, we see that ERPS uses far less time to reach a required accuracy than algorithm 1 does.

6.1.2 Continuous Action Space

We test the algorithm when the action space A is continuous, where the service completion probability can be any value between 0 and 1. Again, two cost functions are considered, corresponding to cases (i) and (ii) in Section 6.1.1. In both cases, the maximum queue length $\mathcal{L}$, state space X, and the discount factor α are all taken to be the same as before.

In the numerical experiments, we approximated the optimal costs J_1^* and J_2^* for each of the respective cases (i) and (ii) by two value functions $\hat{J}_1^*$ and $\hat{J}_2^*$, which were computed by

algorithms	parameters	Avg. time (std err)	actual relerr (std err)
ERPS $r = 10$	$q_0 = 0.0$	13.31 (0.60)	7.63e-07 (3.71e-08)
	$q_0 = 0.1$	1.20 (0.03)	4.99e-07 (5.47e-08)
	$q_0 = 0.3$	0.96 (0.04)	3.26e-07 (4.83e-08)
	$q_0 = 0.5$	0.97 (0.03)	3.84e-07 (5.08e-08)
	$q_0 = 0.7$	1.61 (0.18)	3.47e-07 (4.91e-08)
	$q_0 = 0.9$	4.03 (0.62)	2.33e-07 (4.62e-08)
ALG. 1	$P_m = 0.1,\ P_g = 0.9,\ P_l = 0.1$	62.4 (3.0)	7.61e-07 (3.67e-08)
	$P_m = 0.3,\ P_g = 0.9,\ P_l = 0.1$	33.3 (1.4)	8.42e-07 (2.76e-08)
	$P_m = 0.5,\ P_g = 0.9,\ P_l = 0.1$	26.6 (1.4)	8.35e-07 (2.93e-08)
	$P_m = 0.7,\ P_g = 0.9,\ P_l = 0.1$	22.1 (1.2)	7.88e-07 (3.34e-08)
	$P_m = 0.9,\ P_g = 0.9,\ P_l = 0.1$	20.2 (1.1)	8.44e-07 (2.55e-08)
	$P_m = 1.0,\ P_g = 1.0,\ P_l = 0.0$	17.6 (0.9)	7.67e-07 (4.08e-08)

Table 4: Average time required to reach a precision of at least 1.0e-6 for different algorithms. All results are based on 30 independent replications. The standard errors are in parentheses.

using the adaptive ERPS algorithm under the following parameter settings: population size $n = 10$; stopping control parameter $K = 10$; exploitation probability $q_0 = 0.5$; initial search range $r = \frac{1}{10}$; tolerance $\varepsilon =$ 1e-12 for **case (i)** and $\varepsilon =$ 1e-10 for **case (ii)**; $K_1 = 5$; $K_2 = 5$; $K_3 = 5$; $\gamma = 2$. We performed 200 independent runs of the adaptive ERPS algorithm for each case, and $\hat{J}_1^*$ ($\hat{J}_2^*$) was obtained as the best solution out of the 200 replications.

We set the population size $n = 10$, termination control parameter $K = 10$, and test the ERPS algorithm by using different values of the search range r. The performance of the algorithm is also compared with that of a deterministic policy iteration (PI) algorithm, where we first uniformly discretize the action space into evenly spaced points by using a mesh size h, and then apply the standard PI algorithm on the discretized problem. Tables 5 and 6 give the performances of both algorithms for cases (i) and (ii), respectively. Note that the relative errors are actually computed by replacing the optimal costs with their corresponding approximations in equation (12).

Test results indicate that ERPS outperforms the discretization-based PI algorithm in both cases, not only in computational time but also in solution quality. We observe that the computational time for PI increases by a factor of 2 for each halving of the mesh size, while the time for ERPS increases at a much slower rate.

algorithms	parameters	Avg. time (std err)	mean relerr (std err)
ERPS $(r = \frac{1}{4000})$	$q_0 = 0.25$	2.54 (0.10)	1.92e-12 (3.64e-13)
	$q_0 = 0.50$	2.27 (0.09)	6.41e-13 (7.07e-14)
	$q_0 = 0.75$	2.92 (0.08)	1.92e-13 (2.69e-14)
ERPS $(r = \frac{1}{8000})$	$q_0 = 0.25$	2.61 (0.10)	4.66e-13 (6.03e-14)
	$q_0 = 0.50$	2.91 (0.10)	1.08e-13 (1.59e-14)
	$q_0 = 0.75$	3.05 (0.11)	6.84e-14 (1.03e-14)
ERPS $(r = \frac{1}{16000})$	$q_0 = 0.25$	2.84 (0.09)	1.33e-13 (2.35e-14)
	$q_0 = 0.50$	3.25 (0.10)	3.06e-14 (4.56e-15)
	$q_0 = 0.75$	3.68 (0.10)	1.89e-14 (2.50e-15)
PI	$h = \frac{1}{4000}$	6 (N/A)	7.96e-09 (N/A)
	$h = \frac{1}{8000}$	12 (N/A)	1.72e-09 (N/A)
	$h = \frac{1}{16000}$	23 (N/A)	4.74e-10 (N/A)
	$h = \frac{1}{32000}$	47 (N/A)	9.52e-11 (N/A)
	$h = \frac{1}{128000}$	191 (N/A)	6.12e-12 (N/A)
	$h = \frac{1}{512000}$	781 (N/A)	3.96e-13 (N/A)

Table 5: Comparison of the ERPS algorithm ($n = 10,\ K = 10$) with the deterministic PI algorithm for **case (i)**. The results of ERPS are based on 30 independent replications. The standard errors are in parentheses.

6.2 A Two-Dimensional Queueing Example

The second example, shown in Figure 5, is a slight modification of the first one, with the difference being that now we have a single queue that feeds two independent servers with different service completion probabilities a_1 and a_2. We consider only the continuous action space case. The action to be chosen at each state x is $(a_1, a_2)^T$, which takes value from the set $A = [0, 1] \times [0, 1]$. We assume that an arrival that finds the system empty will always be served by the server with service completion probability a_1. The state space of this problem is $X = \{0, 1_{S_1}, 1_{S_2}, 2, \ldots, 48\}$, where we have assumed that the maximum queue length (no including those in service) is 46, and $1_{S_1}, 1_{S_2}$ are used to distinguish the situations whether server 1 or server 2 is busy when there is only one customer in the system. As before, the discount factor $\alpha = 0.98$.

The one-stage cost is taken to be

$$R(y, a_1, a_2) = y + \left[\frac{|X|}{2}\cos(\pi a_1) - y\right]^2 I_{\{S_1\}} + \left[\frac{|X|}{2}\sin(\pi a_2) - y\right]^2 I_{\{S_2\}},$$

algorithms	parameters	Avg. time (std err)	mean relerr (std err)
ERPS ($r = \frac{1}{4000}$)	$q_0 = 0.25$	2.75 (0.10)	8.49e-11 (1.50e-11)
	$q_0 = 0.50$	2.91 (0.09)	1.76e-11 (2.90e-12)
	$q_0 = 0.75$	3.16 (0.09)	8.53e-12 (1.21e-12)
ERPS ($r = \frac{1}{8000}$)	$q_0 = 0.25$	3.09 (0.12)	1.70e-11 (2.57e-12)
	$q_0 = 0.50$	3.00 (0.12)	4.17e-12 (4.94e-13)
	$q_0 = 0.75$	3.62 (0.08)	1.55e-12 (1.47e-13)
ERPS ($r = \frac{1}{16000}$)	$q_0 = 0.25$	3.20 (0.10)	6.08e-12 (1.17e-12)
	$q_0 = 0.50$	3.28 (0.11)	1.19e-12 (1.40e-13)
	$q_0 = 0.75$	4.20 (0.12)	4.25e-13 (5.05e-14)
PI	$h = \frac{1}{4000}$	6 (N/A)	2.71e-07 (N/A)
	$h = \frac{1}{8000}$	11 (N/A)	5.66e-08 (N/A)
	$h = \frac{1}{16000}$	22 (N/A)	1.58e-08 (N/A)
	$h = \frac{1}{32000}$	43 (N/A)	5.21e-09 (N/A)
	$h = \frac{1}{128000}$	176 (N/A)	3.58e-10 (N/A)
	$h = \frac{1}{512000}$	727 (N/A)	1.71e-11 (N/A)

Table 6: Comparison of the ERPS algorithm ($n = 10,\ K = 10$) with the deterministic PI algorithm for **case (ii)**. The results of ERPS are based on 30 independent replications. The standard errors are in parentheses.

where

$$I_{\{S_i\}} = \begin{cases} 1 & \text{if server } i \text{ is busy,} \\ 0 & \text{otherwise,} \end{cases} \quad (i = 1, 2), \quad \text{and} \quad y = \begin{cases} 1 & \text{if } x \in \{1_{S_1}, 1_{S_2}\}, \\ x & \text{otherwise.} \end{cases}$$

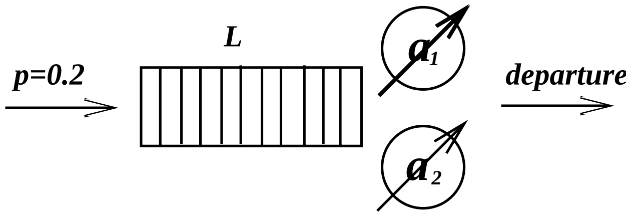


Figure 5: A two-dimensional queueing example.

Again, in computing the relative error, we approximated J^* by $\hat{J}^*$, which was computed by using the adaptive ERPS algorithm under the same settings (e.g., parameter settings, number of replications) as in **case (ii)** of Section 6.1.2. The value of $\|\hat{J}^*\|_\infty$ is approximately 1.72e+04.

algorithms	parameters	Avg. time (std err)	mean relerr (std err)
ERPS ($r = \frac{1}{100}$)	$q_0 = 0.25$	3.26 (0.14)	2.60e-06 (1.36e-07)
	$q_0 = 0.50$	3.20 (0.15)	1.06e-05 (9.17e-06)
	$q_0 = 0.75$	3.64 (0.14)	8.98e-05 (2.54e-05)
ERPS ($r = \frac{1}{200}$)	$q_0 = 0.25$	3.37 (0.12)	6.67e-07 (3.59e-08)
	$q_0 = 0.50$	3.28 (0.12)	9.58e-06 (9.20e-06)
	$q_0 = 0.75$	3.89 (0.17)	9.38e-05 (2.47e-05)
ERPS ($r = \frac{1}{400}$)	$q_0 = 0.25$	3.78 (0.11)	1.50e-07 (8.30e-09)
	$q_0 = 0.50$	3.85 (0.12)	9.30e-06 (9.21e-06)
	$q_0 = 0.75$	4.45 (0.14)	4.59e-05 (1.90e-05)
PI	$h = \frac{1}{100}$	15 (N/A)	1.65e-04 (N/A)
	$h = \frac{1}{200}$	57 (N/A)	4.30e-05 (N/A)
	$h = \frac{1}{400}$	226 (N/A)	8.87e-06 (N/A)

Table 7: A two-dimensional test example. The results of ERPS are based on 30 independent replications ($n = 10,\ K = 10$).

The performances of the ERPS and the discretization-based PI are reported in Table 7. In ERPS, both the population size n and the stopping control parameter K are set to 10. In PI, we adopt a uniform discretization, where the same mesh size h is used in both directions of the action space. Notice that the computational time for PI increases by a factor of 4 for each halving of the mesh size, whereas the time required by ERPS increases much more slowly.

In Table 8, we compare the performance of the adaptive ERPS algorithm and the original ERPS algorithm in obtaining high quality solutions. In both algorithms, we choose the population size $n = 10$, the stopping control parameter $K = 10$, and the exploitation probability $q_0 = 0.5$. In adaptive ERPS, the initial search range $r = 0.1$, $\gamma = 2$, parameters K_1, K_2 and K_3 are all set to 5, and the improvements in elite policies are evaluated in the infinity-norm. We see that in order to obtain more and more accurate solutions, the search range in ERPS has to be chosen excessively small, which causes significant increase in computational effort. In contrast, the adaptive ERPS achieves better solutions within less time; moreover, the algorithm provides us with a rough estimation of the solution quality: as mentioned in Section 5, the average difference between the resultant value function J and the optimal cost J^* (i.e., $\|J - J^*\|_\infty$) will be of the same order of magnitude as ε; and the relative error can also be estimated as:

$$\text{relerr} = \frac{\|J - J^*\|_\infty}{\|J^*\|_\infty} \approx \frac{\|J - \hat{J}^*\|_\infty}{\|\hat{J}^*\|_\infty} \approx \frac{\varepsilon}{\|\hat{J}^*\|_\infty}.$$

algorithms	parameters	Avg. time (std err)	mean relerr (std err)	$\|J - \hat{J}^*\|_\infty$ (std err)
ERPS	$r = \frac{1}{20000}$	16.4 (0.2)	2.25e-11 (8.88e-13)	N/A (N/A)
	$r = \frac{1}{40000}$	24.8 (0.3)	5.04e-12 (1.95e-13)	N/A (N/A)
	$r = \frac{1}{80000}$	39.1 (0.5)	1.02e-12 (7.18e-14)	N/A (N/A)
Adaptive ERPS	ε =1e-07	13.8 (0.7)	9.28e-12 (3.22e-12)	1.59e-07 (5.54e-08)
	ε =1e-08	15.7 (0.8)	3.95e-13 (1.67e-13)	6.80e-09 (2.87e-09)
	ε =1e-09	17.1 (0.7)	1.09e-13 (3.12e-14)	1.87e-09 (5.37e-10)

Table 8: Comparison of ERPS ($n = 10,\ K = 10,\ q_0 = 0.5$) with adaptive ERPS ($n = 10,\ K = 10,\ q_0 = 0.5,\ r = 0.1,\ K_1 = K_2 = K_3 = 5,\ \gamma = 2$), based on 30 independent replications.

7 Conclusions and Future Work

In this paper, we presented an evolutionary, population-based method called ERPS for solving infinite horizon discounted cost MDP problems. We showed that the algorithm converges to an optimal policy w.p.1. We also illustrated the algorithm by applying it to two controlled queueing examples with large or uncountable action spaces. Numerical experiments on these small examples indicate that the ERPS algorithm is a promising approach, outperforming some existing methods (including the standard policy iteration algorithm).

Many challenges remain to be addressed before the algorithm can be applied to realistic-sized problems. The motivation behind ERPS is the setting where the action space is extremely large so that enumerating the entire action space becomes computationally impractical; however, the approach still requires enumerating the entire state space. To make it applicable to large state space problems, the algorithm will probably need to be used in conjunction with some other state space reduction techniques such as state aggregation or value function approximation. This avenue of investigation clearly merits further research.

Another important issue is the dependence of ERPS on the underlying distance metric, as determining a good metric could be challenging for those problems that do not have a natural metric already available. One possible way to get around this is to adaptively updating/changing the *action selection distribution* $\mathcal{P}$ at each iteration of the algorithm based on the sampling information obtained during the previous iterations. This actually constitutes a learning process; the hope is that more promising actions will have larger chances of being selected so that the future search will be biased toward the region containing high quality solutions (policies).

Another practical issue is the choice of the exploitation probability q_0. As noted earlier, the parameter q_0 serves as a tradeoff between exploitation and exploration in action selections. Preliminary experimental results indicate some robustness with respect to the value of this parameter, in that values between 0.25 and 0.75 all seem to work well; however, this may not hold for larger problems or other settings, so further investigation is required. One approach is to design a similar strategy as in simulated annealing algorithms and study the behavior of the algorithm when the value of q_0 is gradually increasing from 0 to 1, which corresponds to the transitioning of the search mechanism from pure random sampling to pure local search.

Acknowledgments

This work was supported in part by the National Science Foundation under Grants DMI-9988867 and DMI-0323220, and by the Air Force Office of Scientific Research under Grants F496200110161 and FA95500410210. We thank the three referees for their detailed comments and suggestions, which have led to a substantially improved paper.

References

Barash, D. 1999. A Genetic Search in Policy Space for Solving Markov Decision Processes. *AAAI Spring Symposium on Search Techniques for Problem Solving under Uncertainty and Incomplete Information, Stanford University, March 1999.*

Bentley, J. 1979. Multidimensional Binary Search Trees in Database Applications. *IEEE Trans. on Software Engineering, 5(4): pages 333–340.*

Bertsekas, D. P. and Castañon, D. A. 1989. Adaptive Aggregation Methods for Infinite Horizon Dynamic Programming. *IEEE Transactions on Automatic Control, Vol. 34, NO. 6.*

Bertsekas, D. P. 1995. *Dynamic Programming and Optimal Control, Volumes 1 and 2.* Athena Scientific.

Chang, H. C., Lee, H. G., Fu, M. C., and Marcus, S. I. 2002. Evolutionary Policy Iteration for Solving Markov Decision Processes. ISR Technical Report TR-2002-31, University of Maryland, College Park. http://techreports.isr.umd.edu/reports/2002/TR-2002-31.

Chavez, E., and Navarro, G. 2000. An Effective Clustering Algorithm to Index High Dimensional Metric Spaces. *Seventh International Symposium on String Processing Information Retrieval (SPIRE'00) September 27 - 29, A Coruña, Spain, page 75.*

Corana, A., Marchesi, M., Martini, C., and Ridella, S. 1987. Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm. *ACM Transactions on Mathematical Software, Vol. 13, No. 3, pages 262–280.*

de Farias, D. P. and Van Roy, B. 2003. The Linear Programming Approach to Approximate Dynamic Programming. *Operations Research, Vol. 51, No. 6, pp. 850–865.*

Demmel, J. W. 1997. *Applied Numerical Linear Algebra.* Society for Industrial and Applied Mathematics.

Guttman, A. 1984. R-trees: A Dynamic Index Structure for Spatial Searching. *In Proc. ACM SIGMOD'84, pages 47–57.*

Lourenco, H. R., Martin, O. C., and Stützle, T. 2001. *Iterated Local Search.* Handbook on MetaHeuristics. Ed. Glover, F., and Kochenberger, G.

MacQueen, J. 1966. A Modified Dynamic Programming Method for Markovian Decision Problems. *J. Math. Anal. Appl., Vol. 14, pp 38–43.*

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley & Sons, Inc.

Rust, J. 1997. Using Randomization to Break the Curse of Dimensionality. *Econometrica, 65, (3), pp. 487–516.*

Srinivas, M., and Patnaik, L. M. 1994. Genetic Algorithms: a survey. *IEEE Computer, Vol. 27, No. 6, pages 17–26.*

Trick, M. and Zin, S. 1997. Spline Approximations to Value Functions: A Linear Programming Approach. *Macroeconomic Dynamics, Vol. 1.*

Tsitsiklis, J. N. and Van Roy, B. 1996. Feature-Based Methods for Large-Scale Dynamic Programming *Machine Learning, Vol. 22, pp. 59–94.*

Wells, C., Lusena, C., and Goldsmith, J. 1999. Genetic Algorithms for Approximating Solutions to POMDPs. Department of Computer Science Technical Report TR-290-99, University of Kentucky. http://cs.engr.uky.edu/ goldsmit/papers/gen.ps